©2022 Published in 10th International Symposium on Innovative Technologies in Engineering and Science 21-23 October 2022 (ISITES2022 Bursa - Turkey)
https://doi.org/10.33793/acperpro.05.03.480

Development An Image Processing Algorithm To Detect Perforated and Non-Perforated Objects

*[1]Fatih Köse and [1]H.Metin Ertunç
*[1]Mechatronics Engineering Department, Kocaeli University, Türkiye

Abstract

In this study, an algorithm was developed that can detect and present number of perforated and non-perforated objects on an image directly or on an image which was captured momentarily by a camera. Firstly, the algorithm was developed using C programming language on Matlab environment and its performance was tested on the artificially produced binary images. At the same time, a developed version of the algorithm was created on Visual Studio Code environment using Python programming language that is used frequently in many software. After successful trials on the prepared artificial images, a real-time object detection algorithm was achieved with inputs of algorithm provided with images taken momentarily by camera. To use the developed algorithm at different applications, Raspberry-Pi based experimental setup was prepared from single board computer platforms and the developed algorithm was run on. In conclusion, it was seen that the number and the exact positions of perforated and non-perforated objects on the images can be detected successfully.

Key words: Image Processing, Raspberry-Pi, Python, Detect Objects

1. Introduction

Digital image processing is a technology that has emerged in 1950s. Since the application cost was high in the early days, this technology could only be implemented in laboratories in developed countries. After academic studies, this technology has also gained commercial meaning and has been used in activities such as production control and quality control applications [1].

Image processing is a process that enables digitization and analysis on a digital image and achieving the desired output with the help of algorithms [2]. Image processing is used in many applications such as changing the size of digital images, detecting objects on the image, face recognition and vehicle recognition systems. In particular, different applications can be seen in security systems and robot applications in the industrial sector, in unmanned aerial vehicle applications in the defense industry, in the health and biomedical fields, in the agriculture sector and many other sectors [3, 4, 5].

Because of developing technology, image processing can be implemented in dynamic environment and changing lighting conditions in real time. Meaningful results can be obtained from raw image data using image processing methods. Recently, graphics processing cards with increasing processing power and development cards introduced as single board computers also increase the usage areas of image processing applications [6, 7].

*Corresponding author: Address: Mechatronics Engineering Department Kocaeli University, 41380, Kocaeli TURKEY. E-mail address: fatihkose86@gmail.com, Phone: +902623033098

Object detection algorithms, which are a subject of image processing applications, can contain differences according to application areas. The algorithm developed to solve the problem of overlapping objects in an image and scattering disparities of the same object can contribute to object identification applications in the next step [8].

One or more of the physical features of the object can be used in object detection algorithms. There are object detection algorithms that can distinguish certain colored objects in an image independent of the shape and size of the object [9].

In this study, an algorithm that can detect perforated and non-perforated objects on the image has been developed and tested on an experimental setup created with a single-board computer. The aim of the study is not only to detect objects with holes, double holes and non-holes but also to determine the numbers of those objects.

2. Materials and Method

2.1. Logic of Algorithm

The algorithm logic developed for the detection of perforated and non-perforated objects basically consists of the stages as follows: binarizing the image, labeling and decoding. The flowchart showing these stages has been presented in Figure 1.

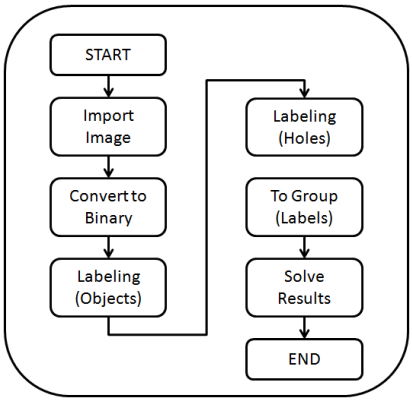

Figure 1. Flowchart of the algorithm

2.1.1. Import Image

In this study, the images used as input were transferred into the algorithm with two different ways. Firstly, studies were carried out on artificial images created in the computer environment. In the next stage of the study, a snapshot was taken on real objects with the help of a camera, and the image was transferred to the algorithm.

2.1.2. Convert Image to Binary

A series of filters are applied before starting the operations on RGB images taken from the camera or images with artificially created gray tones. The image taken from as RGB from the

camera is first converted to a grayscale image. After this image is converted to matrix format, a 2D matrix is obtained by filtering the matrix sizing. Then, the value of each pixel on the matrix is compared with a previously determined threshold value and the grayscale image is arranged that consist of black and white pixels. Thus, the resulting matrix consists of only 0s and 1s. This logic is carried out by the operations given in Equation 1 and Equation 2.

$$I(x,y) < Th \text{ so } \; I(x,y) = 0 \tag{1}$$

$$I(x,y) \geq Th \text{ so } \; I(x,y) = 1 \tag{2}$$

In the equations; I denotes a matrix of size $M \times N$ in which the image is transformed, x and y represent the element of the matrix in horizontal and vertical coordinates, respectively ((x,y)'th pixel if considered as an image), and Th denotes the threshold value. In this study, $Th = 127$ was determined. Each pixel can take a value between 0 to 255.

In the new image that is formed as black (0) and white (1), 1s represent objects and 0s represent the ground. In order to minimize or eliminate the noise in the obtained image, opening and closing operations, which include morphological operations, are applied and the noise are filtered.

2.1.3. Labeling Objects

In the image shown in Figure 2, black areas represent the ground, i.e. values with 0, and white areas represent *A* and *B* objects, i.e. values with 1. As seen in the figure, while object *A* is full, there is a hole in object *B*. As *M* is the number of rows and *N* is the number of columns are define matrix size.

Before labeling an image as in Figure 2, the image is expanded by one pixel from each edge in order to be able to process it. This expansion is done using by vectors consisting of 0s. The enlarged image is presented in Figure 3.

As seen in Figure 2, the starting point indicated by the green arrows has remained in the same place in Figure 3 after the expansion process. The location of the starting point is important for the next operations to be applied in the algorithm.

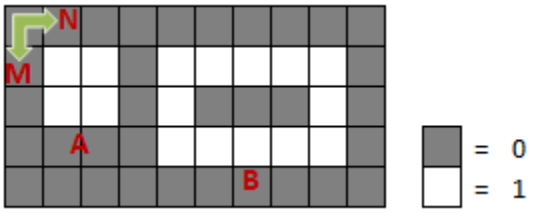

Figure 2. A and B objects

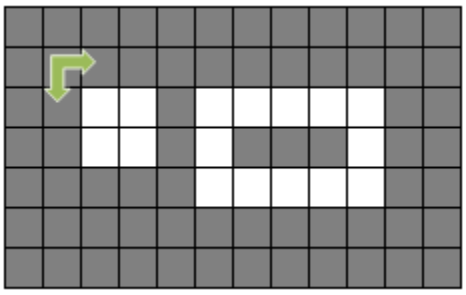

Figure 3. Input image

For labeling process, the neighborhoods of the 1s are checked. In the image matrix, the elements in each row and column are scanned sequentially. If the value of the element is 0, 0 is written to the (x,y)'th element in the label matrix. If the value of the element in the image matrix is 1, the following steps are applied.

Step 1. Increase the label counter by 1 if the element values to the left and above of the related element are 0.
Step 2. Use the previous label value if the element values to the left and above of the related element are 1.
Step 3. If the element value above or to the left of the related element is 1, the label counter remains the same.

The value of label counter that takes value according to above logical approach is applied until the scanning process in the image matrix is finished by writing to the (x,y)'th element in the label matrix. The number of elements in the image matrix and the label matrix are equal. Thus, a labeling matrix is created for the detection of objects in the image matrix. In Figure 4, the labeling process has been shown according to the logical approach presented above. Here all of the black segments are labeled 0, but for understandable only the hole interiors in object *B* are labeled 0. As seen, both of labels 1 and 2 have been created in objects *A* and *B*.

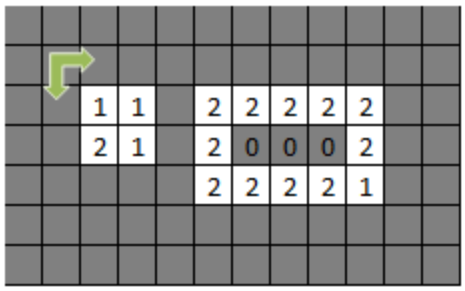

Figure 4. Labeling objects

2.1.4. Labeling Holes

After the label matrix of the objects is created, a label matrix is similarly created to detect the holes. The key point here is the process of distinguishing the pixels that make up the holes from the pixels of the non-object surfaces in the picture. For this, the neighborhood of 0s in the

algorithm is checked. Not only the image matrix is used, but also it is decided by looking at the states in the label matrix and the label matrix is updated each time. If the element value in the image matrix is 1, in other words, if it is over an object pixel, the counter value is not changed and 1 is written to the (x,y)'th element in the label matrix. If the element value in the image matrix is 0, the following steps are applied.

Step 1. Increment the label counter by 1 if the element values to the left and above of the related element are 1.

Step 2. If the value of the element above and to the left of the related element in the label matrix is greater than 1, the label counter remains the same.

Step 3. If the value to the left of the related element in the image matrix is 1 and the value of the element above it in the label matrix is greater than 1, then the label counter will remain the same.

Step 4. If the value to the left of the related element in the label matrix is greater than 1 and the value of the element above it in the image matrix is 1, then the label counter will remain the same.

Step 5. If the element values to the left and above of the related element in the image matrix are 0, set the related element to 0 in the label matrix. Keep the label counter the same.

According to the above logical approach, the value of the label counter that takes value is written to the (x,y) element in the label matrix. This process continues until the scanning process in the image matrix reaches the last element. In Figure 5, the labeling matrix prepared according to the logical approach presented above is shown. As seen in the figure, the values of the ground pixels are labeled as 0, the values of the object pixels are labeled as 1, and the values of the perforated region are labeled as 2. Although the ground and the perforated area were the same color, separation was carried out in the labeling process. The value of the label counter was started from 2 so that the values in the hole did not interfere with the value of 1 given to the objects.

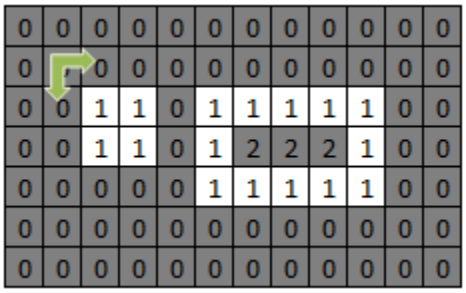

Figure 5. Labeling holes

After this step, which labels the holes, unwanted pixels, namely 1' values, occurred in the lower right corners of objects in different geometries. In order to filter this situation, the following operations are added to the algorithm. While the scanning process was started from the top left of the matrix in the previous sections, the scanning process for this section was started from the bottom right corner of the matrix.

Step 1. If the value to the left of the related element in the label matrix is 0 and the value of the element above it is greater than 1, set the element above the related element to 0.

2.1.5. To Group Labels

With the labeling process, objects and, if any, holes in the object are labeled. Many labels may have been used to describe an object. When looking at object *A* in the label matrix in Figure 4, two different labels (numbers 1 and 2) were used to identify a single object. In object *B*, it is continued from a single label, but the value of the label (2) is the same as the value in object *A*, and since it is an object with holes, there are also 0s in the object.

In order to have a single label value for each object in the matrix, a scanning process is performed in the label matrix where the objects are labeled to give sequential label values to the objects from left to right and from up to down. If the related element value is 1 or greater than 1, the following steps are applied.

Step 1. Generate a 3×3 matrix consisting of neighbors with the related element in the center.
Step 2. Take the minimum of the generated matrix (excluding 0) and assign it to a variable.
Step 3. In the generated matrix, replace the value of elements of 1 and greater than 1 with the value kept in the variable.
Step 4. Put the new values in the generated matrix in their places with the related element in the label matrix in the center.

The above steps apply to the elements in the entire label matrix. In Figure 6, a visual of the applied steps for object *A* is presented. As seen, a 3×3 matrix has been created. The white regions represent the labeled elements of the *A* object, the black parts the ground region around the object, and the red region the element in the center of the matrix at the time of scanning. As the above four steps are followed, the scanning process moves one right. It is seen that the labels with a value of 2 have been adjusted to 1 when all the steps and scanning process were finished in object *A*. The same is done in object *B* and the labeling matrix presented in Figure 4 takes the values in Figure 7. In this way, the labels of each object are collected in a single value. The grouping logic was similarly applied in the matrix where the holes were labeled.

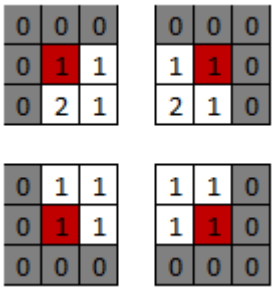

Figure 6. Generated 3×3 matrix

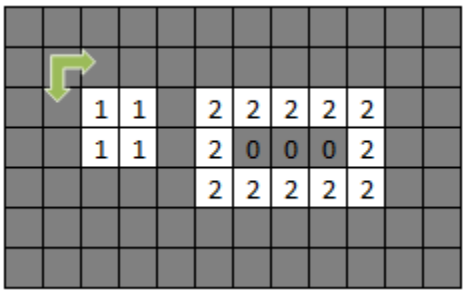

Figure 7. Grouped labels

2.1.6. To Solve Results

After grouping the labels, two different matrices which holes and objects are labeled are obtained. The values in the matrix where the objects are kept are assigned to an array except 0. When the size of this array is taken, the number of all objects in the image is found.

Since the labels of the holes are started from 2, the process of throwing into the array includes values greater than 1. If the size of this array is taken, the number of perforated objects in the image is found. The found values are presented to the user on an interface.

3. Experimental Results and Discussions

3.1. Experimental Setup

Raspberry-Pi Model B card has been used in the experimental setup, and Thonny Python, which is associated to the card with the operating system, has been used as the IDE. A wi-fi receiver has been added to this card in order to download the libraries needed as hardware. The setup has been completed with a standard keyboard, mouse and monitor. All codes have been written in Python. In Figure 8, the photograph of the experimental setup has been presented.

In the experiments on the card, artificially prepared images were given as input to the algorithm. In experiments performed on real images with the help of a camera, the algorithm was compiled and run on the Visual Studio Code program on the computer.

3.2. Experimental Results

Firstly, artificial images were given as input to the algorithm. The performance of the algorithm was observed by adding single-hole and double-hole objects to the same image. Artificially given images and their results are presented in Figure 9 and Figure 10. The visuals of the experiments performed with the camera are as in Figure 11 and Figure 12.

As can be seen in the algorithm outputs, the objects are correctly detected and presented to the user.

Figure 8. Experimental setup

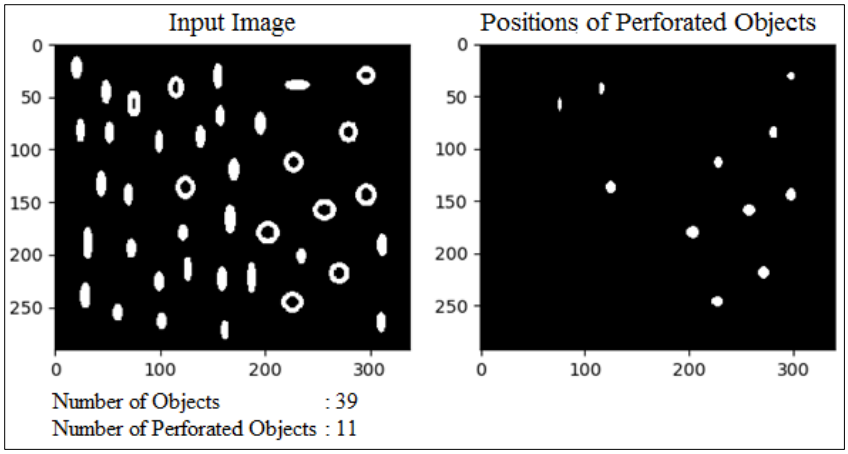

Figure 9. Artificial input image-1and results

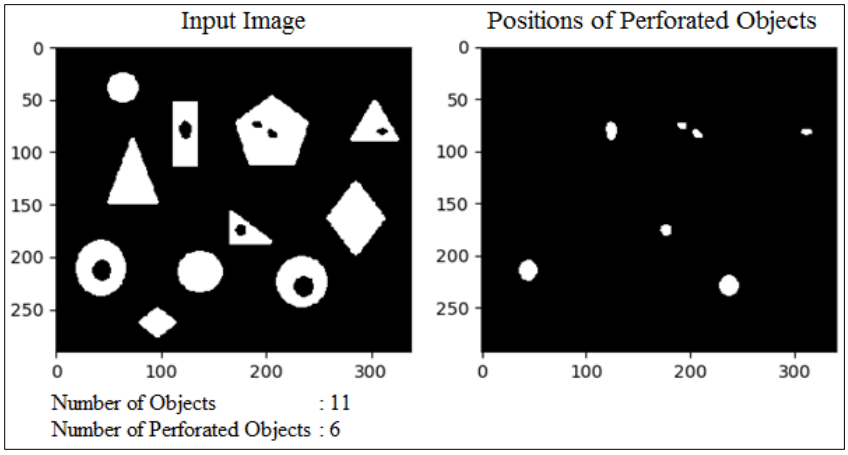

Figure 10. Artificial input image-2 with double-hole object and results

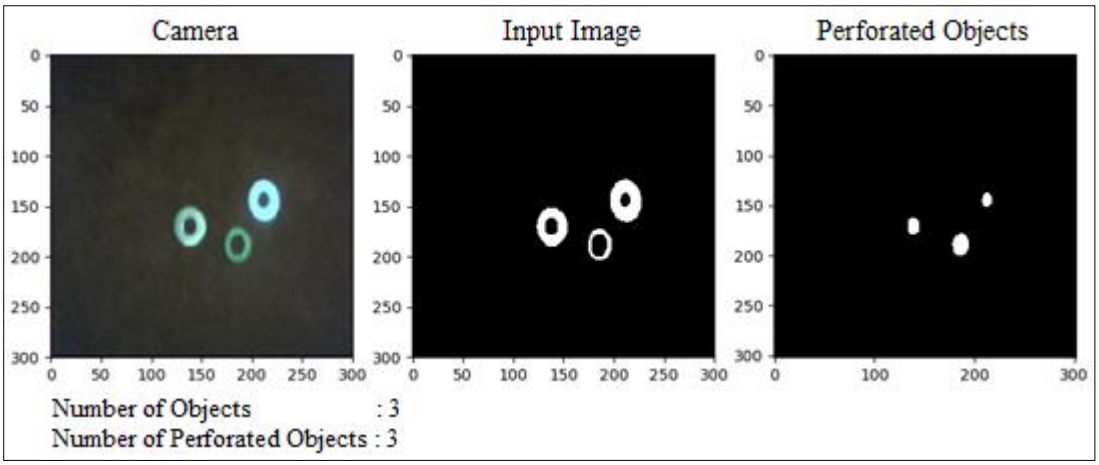

Figure 11. Input image-1 taken by camera and results

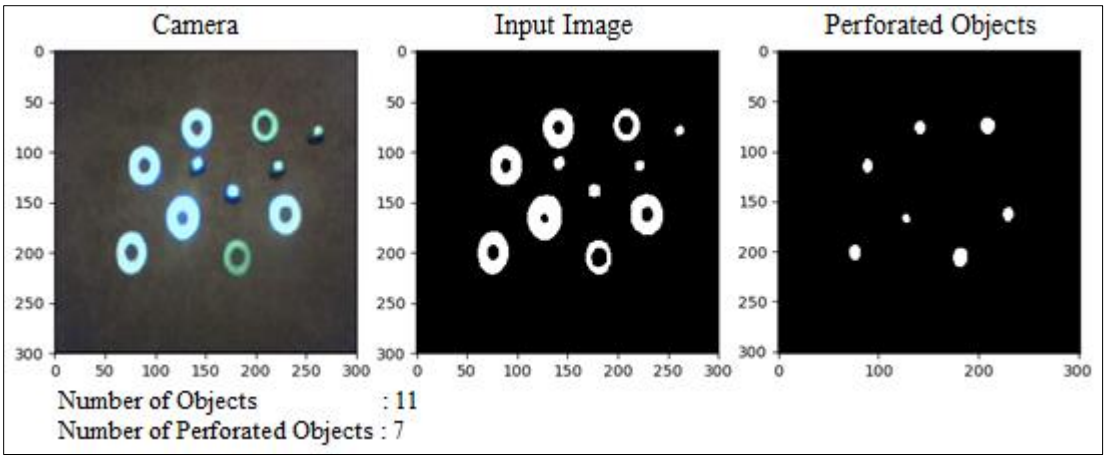

Figure 12. Input image-2 taken by camera and results

Conclusions

In this study, an algorithm that can be used in the detection of perforated and non-perforated objects has been developed and tested. Evaluating at the experimental results, it is seen that the algorithm successfully detects objects with holes, double holes and non-holes and successfully gives their numbers to the user. Morever the study was successfully run on a single-board computer, and a basis was created for different applications.

While there is no problem in taking artificial images as input to the algorithm, many parameters affect the quality of the instantaneous image captured by a camera. Homogeneous distribution of light in the environment, ground color, position of the camera are some of these factors that affect the quality. In order to reduce the effects of these factors, software filtering precautions were taken and the success rate of the algorithm was increased in the study.

References

[1] K. Yaman, N. Aktürk. Görüntü işleme ile kişi yoğunluklarının belirlenmesi. 10. Ulusal Makine Teorisi Sempozyumu 2001.

[2] İ. Soyhan, S. Gürel, S.A. Tekin. Yapay zeka tabanlı görüntü işleme tekniklerinin insansız hava araçları üzerinde uygulamaları. European Journal of Science and Technology Special Issue 2021; 24:473-469.

[3] A. Eldem, H.Eldem, A. Palalı. Görüntü işleme teknikleriyle yüz algılama sistemi geliştirme. BEU Journal of Science 2017; 6(2):48-44.

[4] A.E. Aytan, Y. Öztürk, E.K. Örgev. Görüntü işleme. İ.Ü. Diş Hekimliği Fakültesi Dergisi 1993; 27(4):277-273

[5] K. Hanbay, H. Üzen. Nesne tespit ve takip metotları: Kapsamlı bir derleme. Turkish Journal of Nature and Science 2017; 6(2):49-40.

[6] L.A. Szolga. On flight real time image processing by drone equipped with raspberry pi4. International Symposium for Design and Technology in Electronic Packaging 2021; p. 334-337.

[7] M. Ariyanto, I. Haryanto, J.D. Setiawan, M. Munadi, M.S. Radityo. Real-time image processing method using raspberry pi for a car model. International Conference on Electric Vehicular Technology 2019; p. 46-51.

[8] C.C. Chiu, W.C. Lo. An object detection algorithm with disparity values.4th International Conference on Imaging, Signal Processing and Communations 2020; p. 20-23.

[9] A. Alexander, M.M. Dharmana. Object detection algorithm for segregating similar coloured objects and database formation. International Conference on Circuits Power and Computing Technologies 2017; p. 1-4.